

ISIS-11 PL/M-86 V2.0 COMPILATION OF MODULE TYPE
 OBJECT MODULE PLACED IN TYPE.OBJ
 COMPILER INVOKED BY: :F0: TYPE.PLM XREF OPTIMIZE(3) DEBUG

```

1  $title ("TYPE utility: Types file to Console")
    type:
    do:

    $include (:f2:copyrt.lit)

    /*
    Copyright (C) 1983
    Digital Research
    P.O. Box 579
    Pacific Grove, CA 93950
    */

    $include (:f2:vaxcmd.lit)

    /***** VAX commands for generation - read the name of this program
    for PROGNAME below.

    $ util := PROGNAME
    $ ccpmsetup          I set up environment
    $ assign "f2:directory()" fl:          I use local dir for temp files
    $ plm86 "util".plm xref "pl" optimize(3) debug
    $ link86 f2:scd.obj, "util".obj to "util".lnk
    $ loc86 "util".lnk od(sm(code,dats,data,stack,const)) -
      ad(sm(code(0),dats(10000h))) ss(stack(+32)) to "util".
    $ h86 "util"

    **** Then, on a micro:
    A>vax progname.h86 $fans
    A>gencmd progname data1b10001

    **** Notes: Stack is increased for interrupts. Const(ants) are last
    to force hex generation.
    ****/

    /*
    Revised:
    19 Jan 80 by Thomas Rolander (mp/m 1.1)
    21 July 81 by Doug Huskey (mp/m 2.0)
    6 Aug 81 by Danny Horovitz (mp/m-86 2.0)
    23 Jun 82 by Bill Fitler (ccp/m-86)
    25 Jun 83 by Fran Borda & Bill Fitler (ccp/m-86 2.0)
    */

    /* MODIFICATION LOG:
    * July 82 whf: abort if wildcard char in filename.
    * Jan 83 fmb : check for openfile error codes in All reg.
    */

    $include (:f2:vermpm.lit)
  
```

CCP/M-86 2.0 V.5
 COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # CC-104

```

=
= /* This utility requires MP/M or Concurrent function calls */
=
= /***** commented out for CCP/M-86 :
= declare Ver$OS literally "11h",
= Ver$Needs$OS literally ""Requires MP/M-d6"";
= *****/
=
2 1 = declare Ver$OS literally "14h",
= Ver$Needs$OS literally ""Requires Concurrent CP/M-86"";
=
=
3 1 = declare Ver$Mask literally "0fdh"; /* mask out Is_network bit */
=
=
4 1 = declare Ver$BDOS literally "30h"; /* minimal BDOS version reqd */
=

```

```

5 1 declare
    true    literally "OFFn",
    false   literally "0",
    forever literally "while true",
    lit     literally "literally",
    proc    literally "procedure",
    dcl     literally "declare",
    addr    literally "address",
    cr      literally "13",
    lf      literally "10",
    ctrlc   literally "3",
    ctrlx   literally "18h",
    bksp    literally "8";

```

```

/*****
*
*      B D O S   I N T E R F A C E
*
*****/

```

```

6 1 mon1:
    procedure (func,info) external;
7 2     declare func byte;
8 2     declare info address;
9 2     end mon1;

10 1 mon2:
    procedure (func,info) byte external;
11 2     declare func byte;
12 2     declare info address;
13 2     end mon2;

14 1 mon3:
    procedure (func,info) address external;
15 2     declare func byte;
16 2     declare info address;
17 2     end mon3;

```

COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

18 1      mon4: procedure(f,a) pointer external;
19 2      declare f byte, a address;
20 2      end mon4;

```

```

21 1      declare cmdrv      byte      external; /* command drive */
22 1      declare fcb (1)    byte      external; /* 1st default fcb */
23 1      declare fcol6 (1)  byte      external; /* 2nd default fcb */
24 1      declare pass0      address   external; /* 1st password ptr */
25 1      declare len0      byte      external; /* 1st passwd length */
26 1      declare pass1      address   external; /* 2nd password ptr */
27 1      declare len1      byte      external; /* 2nd passwd length */
28 1      declare tbuff (1)  byte      external; /* default dma buffer */

```

```

/*****
*
*      B O O S      Externals
*
*****/

```

```

29 1      read$console:
30 2          procedure byte;
31 2          return mon2 (1,0);
32 2      end read$console;

32 1      printchar:
33 2          procedure (char);
34 2              declare char byte;
35 2              call mon1 (2,char);
36 2      end printchar;

36 1      conin:
37 2          procedure byte;
38 2          return mon2(6,0fdh);
39 2      end conin;

39 1      print$buf:
40 2          procedure (buff$adr);
41 2              declare buff$adr address;
42 2              call mon1 (9,buff$adr);
43 2      end print$buf;

43 1      check$con$stat:
44 2          procedure byte;
45 2          return mon2(11,0);
46 2      end check$con$stat;

46 1      version: procedure address;
47 2          /* returns current cp/m version & */
48 2          return mon3(12,0);
49 2      end version;

49 1      con$status:
50 2          procedure byte;

```

COPYRIGHT © 1981, 1982, 1983

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

50 2      return mon2 (11,0);
51 2      end con$status;

52 1      open$file:
53 2          procedure (fcb$address) address;
54 2              declare fcb$address address;
55 2              return mon3 (15,fcb$address);
56 2          end open$file;

57 1      close$file:
58 2          procedure (fcb$address) byte;
59 2              declare fcb$address address;
60 2              return mon2 (16,fcb$address);
61 2          end close$file;

62 1      read$record:
63 2          procedure (fcb$address) byte;
64 2              declare fcb$address address;
65 2              return mon2 (20,fcb$address);
66 2          end read$record;

67 1      setdma: procedure(dma);
68 2          declare dma address;
69 2          call mon1(26,dma);
70 2          end setdma;

71 1      /* Off => return 8005 errors */
72 1      return$errors:
73 2          procedure(mode);
74 2              declare mode byte;
75 2              call mon1 (45,mode);
76 2          end return$errors;

77 1      terminate:
78 2          procedure;
79 2              call mon1 (143,0);
80 2          end terminate;

81 1      declare
82 2          parse$fn      structure (
83 2              buff$adr  address,
84 2              fcb$adr   address);
85 2          /* The input to parsefilename */

86 1      declare (saveax,savecx) word external; /* reg return vals, set in mon1 */

87 1      parse: procedure;
88 2          declare (retcode,errcode) word;

89 2          call mon1(152,.parse$fn);
90 2          retcode = saveax;
91 2          errcode = savecx;
92 2          if retcode = 0ffffh then      /* parse returned an error */
93 2              do;
94 3                  call print$buf(."Invalid Filespec$");
95 3                  if errcode = 23 then call print$buf(." (drive)$");
96 3                  else if errcode = 24 then call print$buf(." (filename)$");
97 3                  else if errcode = 25 then call print$buf(." (filetype)$");
98 3                  else if errcode = 38 then call print$buf(." (password)$");

```

COPYRIGHT © 1981, 1982, 1983

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

95 3      call print$buf(('%',13,10,'%')); call terminate;
96 2      end;
      end parse;

```

```

/*****
*
*      S U B R O U T I N E S
*
*****/

```

COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

97 1      /* upper case character from console */
98 2      crlf:  proc;
99 2          call printchar(cr);
100 2         call printchar(lf);
          end crlf;
/* ----- */

```

```

          /* fill string @ s for c bytes with f */
101 1      fill:  proc(s,f,c);
102 2          dcl s addr;
          (f,c) byte;
          a based s byte;

103 2          do while (c:=c-1)<>255;
104 3              a = f;
105 3              s = s+1;
106 3          end;
107 2          end fill;
/* ----- */

```

```

          /* upper case character from console */
108 1      ucase:  proc byte;
109 2          dcl c byte;

110 2          if (c:=conin) >= 'a' then
111 2              if c < '(' then
112 2                  return(c-20h);
113 2          return c;
114 2          end ucase;
/* ----- */

```

```

          /* get password and place at fcb + 16 */
115 1      getpasswd:  proc;
116 2          dcl (i,c) byte;

117 2          call crlf;
118 2          call crlf;
119 2          call print$buf(('%Password ? ', '%'));
120 2      retry:
          call fill(.fcb16, ' ', 8);

```

```

121 2      do i = 0 to 7;
122 3      nxtchr:
123 3          if (c:=ucase) >= " " then
124 3              fcb16(1)=c;
125 3          if c = cr then
126 3              goto exit;
127 3          if c = ctrlx then
128 3              goto retry;
129 3          if c = bksp then do;
130 4              if i<1 then
131 4                  goto retry;
132 4              else do;
133 5                  fcb16(1:=i-1)=" ";
134 5                  goto nxtchr;
135 5              end;
136 4          end;
137 3          if c = 3 then
138 3              call terminate;
139 3          end;
140 2      exit:
141 2          c = check$con$stat;
142 2          end getpasswd;

142 1      /*****
143 1          *
144 1          *      M A I N   P R O G R A M
145 1          *
146 1          *****/

147 1      declare (eod,i,char) byte;
148 1      declare control$z literally "1AH";

149 1      /*
150 1          Main Program
151 1      */

152 1      declare (cnt,tcnt) byte;
153 1      declare (ver, error$code) address;

154 1      declare last$dseg$byte byte
155 1          initial (0);

156 1      plm$start: procedure public;
157 2          ver = version;
158 2          if low(ver) < Ver$BDOS or (high(ver) and Ver$Mask) = 0 then do;
159 3              call print$buf (.(Ver$Needs$OS,"$"));
160 3              call mon1(0,0);
161 3          end;

162 2          tcnt,
163 2          cnt = 0;
164 2          if fcb16(1) = "P" then
165 2              do;
166 3              if fcb16(2) = " " or fcb16(2) = "A" then
167 3                  cnt = 24;
168 3              else

```

COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

159 3      cnt = (fcb16(2)-'0')*10
          + (fcb16(3)-'0');
160 3      end;

161 2      parse$fn.buff$adr = .tbuff(1);
162 2      parse$fn.fcb$adr = .fcb;
163 2      call parse;

164 2      do i = 1 to 11;                      /* check for wildcards */
165 3          if fcb(i) = '?' then do;
166 4              call print$buf(.( "No wildcards allowed.", "i"));
167 4              call crlf;
168 4              call terminate;
169 4              end;
170 4          end;
171 3      end;

172 2      call return$errors(0FFh);              /* return after error message */
173 2      call setdma(.fcb16);                  /* set dma to password */
174 2      fcb(6) = fcb(6) or 80h;                /* open in RO mode */
175 2      error$code = open$file(.fcb);
176 2      if low(error$code) = 0FFh then
177 3          do;
178 4              if high(error$code) = 0 then
179 5                  do;
180 6                      call print$buf(.( "File not found.", "i"));
181 6                      call terminate;
182 6                      end;
183 4              else if high(error$code) = 7 then      /* User left out password */
184 5                  do;
185 6                      call getpasswd;
186 6                      call crlf;
187 6                      call setdma(.fcb16);          /* set dma to password */
188 6                      fcb(6) = fcb(6) or 80h;        /* open in RO mode */
189 6                      error$code = open$file(.fcb);
190 6                      end;
191 4              else if high(error$code) = 4 then
192 5                  do;
193 6                      call crlf;
194 6                      call print$buf(.( "Invalid Filespec.", "i"));
195 6                      end;
196 3              end;                      /* Error Checks */
197 2      if low(error$code) <> 0FFh then
198 3          do;
199 4              call return$errors(0);          /* reset error mode */
200 4              call setdma(.tbuff);
201 4              fcb(32) = 0;
202 4              eod = 0;
203 4              do while (not eod) and (read$record(.fcb) = 0);
204 5                  do i = 0 to 127;
205 6                      if (char := tbuff(i)) = control$z
206 7                          then eod = true;
207 5                      if not eod then
208 6                          do;

```

COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
        call terminate;
    end;

****/
209 6      if cnt <> 0 then
210 6      do;
211 7          if char = 0ah then
212 7          do;
213 8              if (tcnt:=tcnt+1) = cnt then
214 8              do;
215 9                  tcnt = read$console;
216 9                  tcnt = 0;
217 9              end;
218 8          end;
219 7      end;
220 6      call printchar (char);
221 6      end;
222 5      end;
223 4      end;

/*
    call close (.fcb);
    *** Warning ***
    If this call is left in, the file can be destroyed.
*/
224 3      end;
225 2      call terminate;
226 2      end plm$start;

227 1      end type;
```

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
102	0000H	1	A. BYTE BASED(S) 104
18	0000H	2	A. WORD PARAMETER 19
5			ADDR LITERALLY 102
5			BACKSP LITERALLY 128
75	0000H	2	BUFFADR. WORD MEMBER(PARSEFN) 161
39	0004H	2	BUFFADR. WORD PARAMETER AUTOMATIC 40 41
109	000CH	1	C. BYTE 110 111 112 113
101	0004H	1	C. BYTE PARAMETER AUTOMATIC 102 103
116	000EH	1	C. BYTE 122 123 124 126 128 137 140
142	0011H	1	CHAR BYTE 205 211 220
32	0004H	1	CHAR BYTE PARAMETER AUTOMATIC 33 34
43	0041H	15	CHECKCONSTAT PROCEDURE BYTE STACK=0008H 140
56	007EH	16	CLOSEFILE. PROCEDURE BYTE STACK=000AH
21	0000H	1	CMOVR. BYTE EXTERNAL(C)
144	0012H	1	CNT. BYTE 154 158 159 209 213
36	0022H	15	CONTIN. PROCEDURE BYTE STACK=0008H 110
49	005FH	15	CONSTATUS. PROCEDURE BYTE STACK=0008H
143			CONTROLZ LITERALLY 205
5			CR LITERALLY 98 124
97	0135H	17	CRLF. PROCEDURE STACK=000EH 117 118 168 186 193
5			CTRLC. LITERALLY
5			CTRLX. LITERALLY 126
5			DCL. LITERALLY
64	0004H	2	DMA. WORD PARAMETER AUTOMATIC 65 66
142	000FH	1	EOD. BYTE 202 203 206 207
78	0006H	2	ERRCODE. WORD 81 85 87 89 91
145	000AH	2	ERRORCODE. WORD 175 176 178 183 189 191 197
140	0205H		EXIT LABEL 125
18	0000H	1	F. BYTE PARAMETER 19
101	0006H	1	F. BYTE PARAMETER AUTOMATIC 102 104
5			FALSE. LITERALLY
22	0000H	1	FCB. BYTE ARRAY(1) EXTERNAL(S) 162 165 174 175 188 189 201 203
23	0000H	1	FCB16. BYTE ARRAY(1) EXTERNAL(C) 120 123 133 155 157 159 173 187
52	0004H	2	FCBADDRESS WORD PARAMETER AUTOMATIC 53 54
60	0004H	2	FCBADDRESS WORD PARAMETER AUTOMATIC 61 62
56	0004H	2	FCBADDRESS WORD PARAMETER AUTOMATIC 57 58
75	0002H	2	FCBADR WORD MEMBER(PARSEFN) 162
101	0146H	32	FILL PROCEDURE STACK=0008H 120
5			FOREVER. LITERALLY
6	0000H	1	FUNC BYTE PARAMETER 7
14	0000H	1	FUNC BYTE PARAMETER 15
10	0000H	1	FUNC BYTE PARAMETER 11
115	0186H	135	GETPASSWD. PROCEDURE STACK=0012H 185
			HIGH BUILTIN 149 178 183 191
142	0010H	1	I. BYTE 164 165 204 205
116	0000H	1	I. BYTE 121 123 130 133
14	0000H	2	INFO WORD PARAMETER 16
10	0000H	2	INFO WORD PARAMETER 12
6	0000H	2	INFO WORD PARAMETER 8

COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

146	0014H	1	LASTOSEG3YTE . . .	BYTE INITIAL														
25	0000H	1	LENO	BYTE EXTERNAL(5)														
27	0000H	1	LEA1	BYTE EXTERNAL(10)														
5			LF	LITERALLY	99													
5			LIT.	LITERALLY														
			LOW.	BUILTIN	149	176	197											
68	0004H	1	MODE	BYTE PARAMETER AUTOMATIC	69	70												
6	0000H		MON1	PROCEDURE EXTERNAL(9) STACK=0000H				34	41	55	70	73	79					
				152														
10	0000H		MON2	PROCEDURE BYTE EXTERNAL(1) STACK=0000H					30	37	44	50	58					
				62														
14	0000H		MON3	PROCEDURE WORD EXTERNAL(2) STACK=0000H					47	54								
18	0000H		MON4	PROCEDURE POINTER EXTERNAL(3) STACK=0000H														
122	01AFH		NXTCHR	LABEL	134													
52	006EH	16	OPENFILE	PROCEDURE WORD STACK=0000H			175	139										
77	0000H	101	PARSE	PROCEDURE STACK=0000H		163												
75	0000H	4	PAKSEFN.	STRUCTURE	79	161	162											
24	0000H	2	PASS0.	WORD EXTERNAL(7)														
26	0000H	2	PASS1.	WORD EXTERNAL(9)														
147	0200H	438	PLMSTART	PROCEDURE PUBLIC STACK=0016H														
39	0031H	16	PRINTBUF	PROCEDURE STACK=0000H		84	86	88	90	92	93	119	151					
				167	180	194												
32	000FH	19	PRINTCHAR.	PROCEDURE STACK=0000H		99	99	220										
5			PROC	LITERALLY	97	101	108	115										
29	0000H	15	READCONSOLE.	PROCEDURE BYTE STACK=0008H				215										
60	008EH	16	READRECORD	PROCEDURE BYTE STACK=000AH				203										
78	0004H	2	RETCODE.	WORD	80	82												
120	0196H		RETRY.	LABEL	127	131												
68	00AEH	19	RETURNERRORS	PROCEDURE STACK=000AH		172	199											
101	0008H	2	S.	WORD PARAMETER AUTOMATIC		102	104	105										
76	0000H	2	SAVEAX	WORD EXTERNAL(12)		80												
76	0000H	2	SAVECX	WORD EXTERNAL(13)		81												
64	009EH	16	SETDMA	PROCEDURE STACK=000AH		173	187	200										
28	0000H	1	TBUFF.	BYTE ARRAY(1) EXTERNAL(11)			161	200	205									
144	0013H	1	TCNT	BYTE	154	213	215	216										
72	00C1H	15	TERMINATE.	PROCEDURE STACK=0008H		94	138	169	181	225								
5			TRUE	LITERALLY	206													
1	0000H		TYPE	PROCEDURE STACK=0000H														
108	0166H	32	UCASE.	PROCEDURE BYTE STACK=000CH				122										
145	0008H	2	VER.	WORD	148	149												
4			VERBOOS.	LITERALLY	149													
3			VERMASK.	LITERALLY	149													
2			VERNEEDSUS	LITERALLY	151													
2			VEROS.	LITERALLY														
46	0050H	15	VERSION.	PROCEDURE WORD STACK=0008H				148										

COPYRIGHT © 1981, 1982, 1983
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

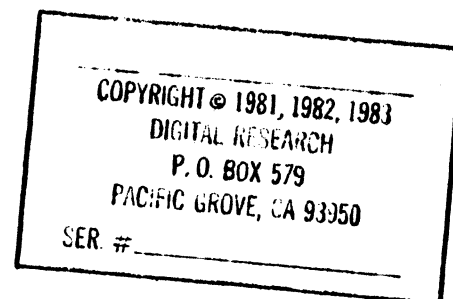
MODULE INFORMATION:

CODE AREA SIZE = 03C3H 963D
 CONSTANT AREA SIZE = 00A2H 162D
 VARIABLE AREA SIZE = 0015H 21D
 MAXIMUM STACK SIZE = 0016H 22D
 422 LINES READ
 0 PROGRAM ERROR(S)

PL/M-86 COMPILER TYPE UTILITY: TYPES FILE TO CONSOLE

PAGE 11

END OF PL/M-86 COMPILATION





ISIS-11 MCS-86 LOCATER, V1.2 INVOKED BY:

#P0: TYPE.LNK DD(SHCODE,DATS,DATA,STACK,CNST)) A)GSM(C03)C00,DATS(10000H)) SSC(STACK+32)) TT TYPE.

SYMBOL TABLE OF MODULE SCO
READ FROM FILE TYPE.LNK
WRITTEN TO FILE TYPE.

BASE	OFFSET	TYPE	SYMBOL
0000H	0300H	PUB	PLMSTART
0000H	0050H	PUB	M0N1
0000H	0050H	PUB	M0N3
1000H	0004H	PUB	BDISK
1000H	0050H	PUB	CM0RV
1000H	0053H	PUB	LEN0
1000H	0056H	PUB	LEN1
1000H	006CH	PUB	FCB16
1000H	0070H	PUB	RR
1000H	0080H	PUB	BUFF
1000H	0080H	PUB	BUFFA
1000H	0100H	PUB	SAVEAX
1000H	0104H	PUB	SAVECX

BASE	OFFSET	TYPE	SYMBOL
0000H	0050H	PUB	X00S
0000H	0050H	PUB	M0N2
0000H	0050H	PUB	M0N4
1000H	0006H	PUB	MAX3
1000H	0051H	PUB	PASS0
1000H	0054H	PUB	PASS1
1000H	005CH	PUB	FCB
1000H	007CH	PUB	CR
1000H	007FH	PUB	RD
1000H	0080H	PUB	T0UFF
1000H	005CH	PUB	FCBA
1000H	0102H	PUB	SAVEBX
1000H	0106H	PUB	SAVEDX

TYPE:	SYMBOLS AND	LINES
1000H	0200H	SYM MEMORY
0000H	010FH	SYM PRINTCHAR
0000H	0122H	SYM CUNIN
STACK	0004H	SYM RUFFADR
0000H	0150H	SYM VERSION
0000H	016EH	SYM OPENFILE
0000H	017EH	SYM CLOSEFILE
0000H	018EH	SYM READRECORD
0000H	019EH	SYM SETDMA
0000H	01AEH	SYM RETURNERRORS
0000H	01C1H	SYM TERMINATE
0000H	01D0H	SYM PARSE
1000H	010EH	SYM ERRCODE
0000H	0246H	SYM FILL
STACK	0006H	SYM F
STACK	0008H	BAS A
1000H	0114H	SYM C
1000H	0115H	SYM T
0000H	0296H	SYM RETRY
0000H	0305H	SYM EXIT
1000H	0118H	SYM T
1000H	011AH	SYM CNT
1000H	0110H	SYM VER
1000H	011CH	SYM LASTDSEGBYTE
0000H	0100H	LIN 29
0000H	010FH	LIN 31
0000H	0112H	LIN 34
0000H	0122H	LIN 36
0000H	0131H	LIN 38
0000H	0134H	LIN 41
0000H	0141H	LIN 43
0000H	0150H	LIN 45
0000H	0153H	LIN 47
0000H	015FH	LIN 49
0000H	016EH	LIN 51
0000H	0171H	LIN 54

0000H	0100H	SYM READCONSOLE
STACK	0004H	SYM CHAR
0000H	0131H	SYM PRINTBUF
0000H	0141H	SYM CHECKCONSTAT
0000H	015FH	SYM CONSTATUS
STACK	0004H	SYM FCBADDRESS
STACK	0004H	SYM FCBADDRESS
STACK	0004H	SYM FCBADDRESS
STACK	0004H	SYM DMA
STACK	0004H	SYM MODE
1000H	0108H	SYM PARSEFN
1000H	010CH	SYM RETCODE
0000H	0235H	SYM CRLF
STACK	0008H	SYM S
STACK	0004H	SYM C
0000H	0266H	SYM UCASE
0000H	0286H	SYM GETPASSWD
1000H	0116H	SYM C
0000H	02AFH	SYM NXTCHR
1000H	0117H	SYM END
1000H	0119H	SYM CHAR
1000H	0118H	SYM TCNT
1000H	0112H	SYM ERRORCODE
0000H	0300H	SYM PLMSTART
0000H	0103H	LIN 30
0000H	010FH	LIN 32
0000H	011EH	LIN 35
0000H	0125H	LIN 37
0000H	0131H	LIN 39
0000H	0130H	LIN 42
0000H	0144H	LIN 44
0000H	0150H	LIN 46
0000H	015FH	LIN 48
0000H	0162H	LIN 50
0000H	016EH	LIN 52
0000H	017EH	LIN 55

COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

0000H	017EH	LIN	56
0000H	018EH	LIN	59
0000H	0191H	LIN	62
0000H	019EH	LIN	64
0000H	01AAH	LIN	67
0000H	01B1H	LIN	70
0000H	01C1H	LIN	72
0000H	01CEH	LIN	74
0000H	01D3H	LIN	79
0000H	01E3H	LIN	81
0000H	01F0H	LIN	84
0000H	01FEH	LIN	86
0000H	020AH	LIN	88
0000H	0216H	LIN	90
0000H	0222H	LIN	92
0000H	0230H	LIN	94
0000H	0235H	LIN	97
0000H	023EH	LIN	99
0000H	0246H	LIN	101
0000H	0255H	LIN	104
0000H	0260H	LIN	106
0000H	0266H	LIN	108
0000H	0273H	LIN	111
0000H	0281H	LIN	113
0000H	0286H	LIN	115
0000H	028CH	LIN	118
0000H	0296H	LIN	120
0000H	02AFH	LIN	122
0000H	02C6H	LIN	124
0000H	02D4H	LIN	128
0000H	02E2H	LIN	133
0000H	02F5H	LIN	137
0000H	02FFH	LIN	139
0000H	030BH	LIN	141
0000H	0310H	LIN	148
0000H	0326H	LIN	151
0000H	0336H	LIN	154
0000H	0347H	LIN	157
0000H	035CH	LIN	159
0000H	0379H	LIN	162
0000H	0382H	LIN	164
0000H	039BH	LIN	167
0000H	03A5H	LIN	169
0000H	03AEH	LIN	172
0000H	03BBH	LIN	174
0000H	03CAH	LIN	176
0000H	03DAH	LIN	180
0000H	03E4H	LIN	182
0000H	03EFH	LIN	185
0000H	03F5H	LIN	187
0000H	0401H	LIN	189
0000H	040DH	LIN	191
0000H	0419H	LIN	194
0000H	042AH	LIN	199
0000H	0437H	LIN	201
0000H	0441H	LIN	203
0000H	0469H	LIN	205
0000H	047FH	LIN	207
0000H	048FH	LIN	211
0000H	04A4H	LIN	215

0000H	01B1H	LIN	68
0000H	01B8H	LIN	69
0000H	019EH	LIN	63
0000H	01A1H	LIN	66
0000H	01AEH	LIN	68
0000H	01B0H	LIN	71
0000H	01C4H	LIN	73
0000H	01D0H	LIN	77
0000H	01D3H	LIN	80
0000H	01E9H	LIN	82
0000H	01F7H	LIN	85
0000H	0203H	LIN	87
0000H	020FH	LIN	89
0000H	021BH	LIN	91
0000H	0229H	LIN	93
0000H	0233H	LIN	96
0000H	0238H	LIN	98
0000H	0244H	LIN	100
0000H	0249H	LIN	103
0000H	025DH	LIN	105
0000H	0262H	LIN	107
0000H	0269H	LIN	110
0000H	027AH	LIN	112
0000H	0286H	LIN	114
0000H	0289H	LIN	117
0000H	028FH	LIN	119
0000H	02A3H	LIN	121
0000H	02B9H	LIN	123
0000H	02C0H	LIN	126
0000H	02DBH	LIN	130
0000H	02F3H	LIN	134
0000H	02FCH	LIN	138
0000H	0305H	LIN	140
0000H	030DH	LIN	147
0000H	0316H	LIN	149
0000H	032DH	LIN	152
0000H	0340H	LIN	155
0000H	0355H	LIN	158
0000H	0373H	LIN	161
0000H	037FH	LIN	163
0000H	038EH	LIN	165
0000H	03A2H	LIN	168
0000H	03A8H	LIN	171
0000H	03B4H	LIN	173
0000H	03C0H	LIN	175
0000H	03D1H	LIN	178
0000H	03E1H	LIN	181
0000H	03E6H	LIN	183
0000H	03F2H	LIN	186
0000H	03FCH	LIN	188
0000H	040BH	LIN	190
0000H	0416H	LIN	193
0000H	0420H	LIN	197
0000H	0430H	LIN	200
0000H	043CH	LIN	202
0000H	045DH	LIN	204
0000H	047AH	LIN	206
0000H	0488H	LIN	209
0000H	0496H	LIN	213
0000H	04AAH	LIN	216

COPYRIGHT © 1981, 1982, 1983

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

0000H 04AFH LIN 220
 0000H 04BCH LIN 223
 0000H 04C1H LIN 226

0000H 04B5H LIN 222
 0000H 043EH LIN 225
 0000H 0100H LIN 227

MEMORY MAP OF MODULE SCD
 READ FROM FILE TYPE.LNK
 WRITTEN TO FILE TYPE.

SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	004C2H	04C3H	G	CODE	CODE
10000H	10107H	0108H	G	DATS	DATA
10108H	1011CH	0015H	W	DATA	DATA
1011EH	10153H	0036H	W	STACK	STACK
10154H	101F5H	00A2H	W	CONST	CONST
10200H	10200H	0000H	G	??SEG	
10200H	10200H	0000H	W	MEMORY	MEMORY

GROUP MAP

ADDRESS	GROUP OR SEGMENT NAME
00000H	CGROUP
	CODE
10000H	DGROUP
	DATS
	STACK
	CONST
	DATA
	MEMORY

